CFT Software: From Here to Reality

Mike Hildreth
University of Notre Dame
D02000 Workshop

My charge:

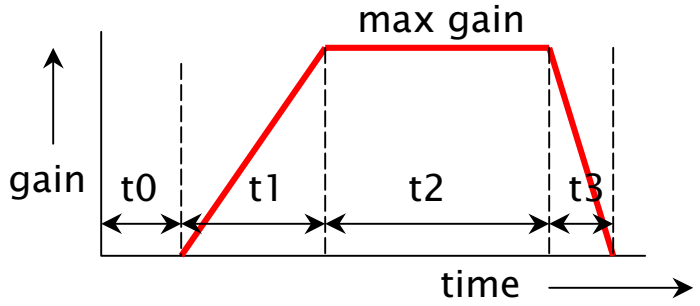
How will the CFT software (reco + sim)

- meet the performance goals?
  - CPU time
  - tracking efficiency vs. Pt
- handle defects such as
  - detector inefficiencies/dead channels
  - noise
  - misalignments

Status of SFT Simulation

The new production release will contain new versions of

- CFTFiberSimulator
  - simulates the correct photon statistics and propagation times given a dE/dx value
- CFTReadoutSimulator
  - Simulates timing structure of SIFT descriminator
  - tunable parameters:



Simple future modifications

It's relative easy to make these routines

- *drop* hits to simulate inefficiencies
  - ⇨ can be done specific to local geometry or randomly
- Add *noise*
- Add *crosstalk*

BUT

Wait until we have **real CFT data** to tune parameters (this fall...)

Until then... a suggestion:

We could provide a MC "post processor" to put these effects into preco4/5 MC samples so that they will be "useful" for comparison to real data.

- (Depends on time scale for large samples vs. CFT commissioning)

★ This avoids generating huge piles of events with the wrong noise/crosstalk/thresholds

➡ whatever we pick now for these parameters WILL be wrong...

Simulating misalignment

Proposal:

The "default" MC CFT geometry should reflect the "as built" detector

- random spread in ϕ-offsets consistent with CMM data
- correct radii for cylinders
- some r-z correlation for axial layers?
- overall rotations/translations removed as they will be when the misalignments are corrected in the data
  - ⇨ could add random spread here, too

Comments welcome!

CFTSim wrap-up

I'm reluctant to define a "default" set of imperfections this instant without knowledge of

- light yields
- FrontEnd electronics thresholds
- true dead-channel count
  - so far, SciFi+waveguides are <0.5%
  - no VLPC info yet

Probably more prudent to save all of the MC hits, add inefficiencies later... (how much later?)

CFTSim wrap-up II

- CPU time optimization in progress
  - Readout simulator still too slow?
  - hope to revisit this soon...

Now, on to reco...

Burning question:

"How badly will CFT Reconstruction fail when detector inefficiencies are added?"

or, to paraphrase

"How many months per event will we need?"

Answer:

We don't know,

and won't for a little while longer...

Still working on trying to get track reconstruction *functional* and *efficient* over the full CFT acceptance

➡ will show "priority list" in a minute...

Required code modifications?

- Effectively, all that is needed to allow inefficiencies in the cluster-finding is to throw "an RCP switch"
  - In practice, new track-finding "paths" need to be defined that allow missing hits
- Code already exists to handle tracks with missing hits
  - "MTracks" already in use (now, #missing hits=0)

Code is ready when we are...

What's the hold-up?

CFT Elements of the "Global Tracking Priority List":

- Establish CFT tracking in the "Overlap Region"
- Understand source of CFT inefficiency for high-Pt tracks with increasing occupancy
- Decrease CPU time per event
- Understand effects of single-hit inefficiencies
- Cosmic Ray tracking for CFT commissioning
- ⋮

Lots to do! We've chosen to arrive at a fully functional and efficient tracking package first, then worry...

Not a non-optimal timeline...

- We estimate that the remaining work on finishing the baseline tracking code will take the remainder of the summer...
- Actual data from the CFT on light yields, thresholds, noise, etc. will start to become available as we start to look at the inefficiency questions
  - ➨ saves optimization of the wrong parameters

- ✸ The only way to speed this up is to have more people working on tracking!

Bottom line on inefficiency question:

We believe that it is crucial to

- have a tracking package that finds tracks within the full tracker acceptance
- understand features/foibles of performance with a "perfect" detector, including CPU time issues

before moving closer to reality.

Obviously, these studies/code development will continue in parallel while we work with "real" data

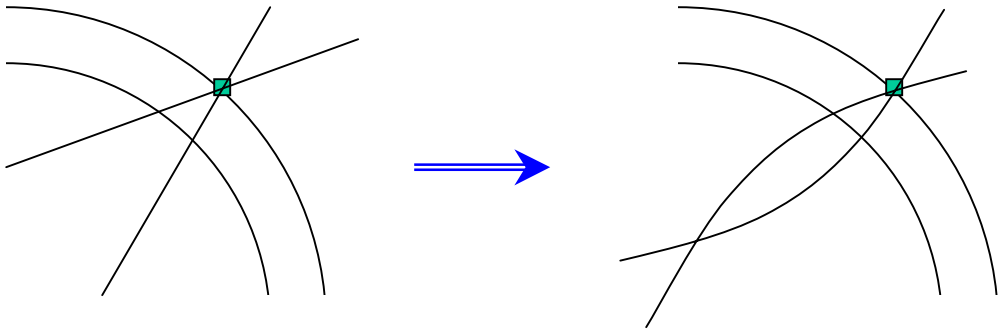
We don't expect huge surprises with added inefficiency

progress on the road to inefficiency

CFT "Overlap" tracking:

New tracking package developed using axial and stereo hits combined into 3-D space points

- new wrinkle: "cluster filters"
  - incorporate geometric/kinematic information when adding new hits to the track
    - i.e. "Does this track point at the origin?"



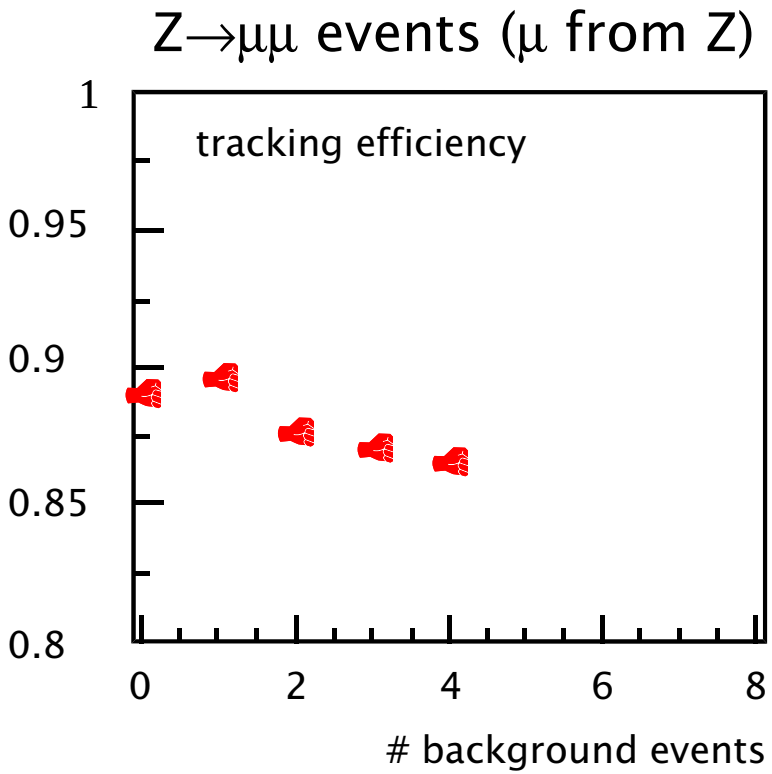
moving towards global zmax, rmin Ptmin knobs

progress on the road to inefficiency

CFT "Overlap" tracking (cont.):

- just now being debugged in CFT central region
  - comparison to original version is possible here
- it does find tracks!
- without limits on geometrical acceptance, CPU time is very large for messy events
  - many ghost hits to deal with
- optimization just beginning, but rapid progress

progress on the road to inefficiency



CFT Inefficiency for high-occupancy events:

- diagnostic tools developed to understand cluster/MC/track assignments
- work in progress - no smoking gun yet, but lots of little "features" are being uncovered...

progress on the road to inefficiency

Decreasing the CPU time per event:

- Not recently an object of intense scrutiny
  - (lots of more pressing things)
- Addition of one (1) new cluster filter at a crucial step speeds up a $Z \rightarrow \mu\mu$ event with 6mb by 50%...
- Our "linear algebra" package is (very?) not optimized for track-fitting-specific problems
  - local optimization (specific vs. generic cases) *will* help, probably a lot!
- Many profiling studies underway

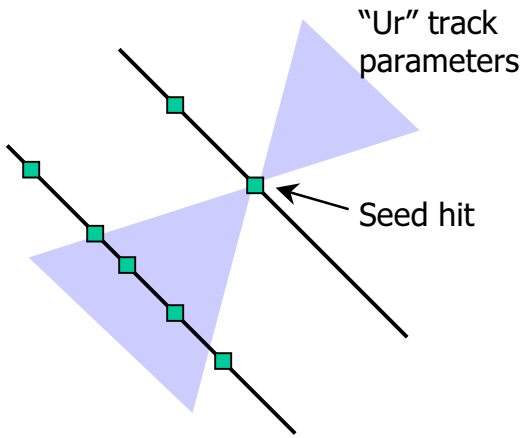
progress on the road to inefficiency

Decreasing the CPU time per event:

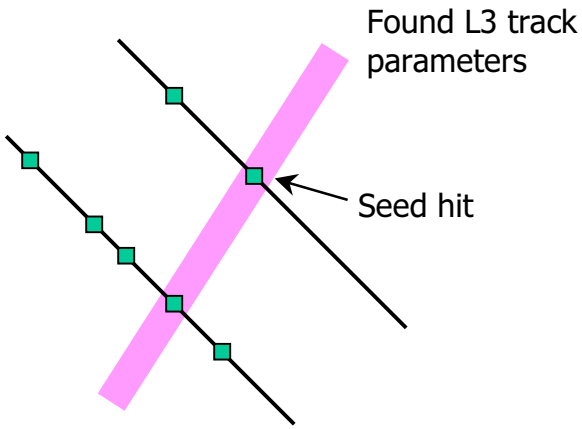
Other ideas:

- We could use the L3 fast tracker as a preprocessor -

Current strategy



Another possibility



alignment studies

- The CMM results will need to be checked with tracks, if for no other reason than to verify that all of the relative signs of the shifts are correct!
- The derived corrections must be included in the geometry description
  - we need to decide with what degree of correctness to treat the observed "wobble" in ϕ as a function of z for the axial layers
  - machinery for updating a cluster position based on its predicted location already exists, could be used for this
- Luckily, "as-built" $\approx$ "perfect"!

Conclusions I:

Functionality of CFT simulation is approaching maturity

- Hit inefficiencies, noise, etc. can be added easily
- Would like to know real rates from CFT data
  - recommend keeping all MC hits for now, tossing later...
- incorporation of "as-built" geometry straightforward
- some tweaking of simulation speed still possible?

Conclusions II

Lots of work still to be done on CFT Reconstruction

- Lots of work *has* been done!
- Recent progress on
  - Overlap tracking
  - efficiency questions
  - CPU optimization

  is encouraging ➨ more new ideas in the pipeline

- Of course, the big question of performance with inefficiencies *has to* and *will* be addressed soon